Assignment 4

Aim:- The aim of this assignment is to explore Naïve Bayes and its application in classification.

Theory:- Naïve Bayes is a supervised learning algorithm used for classification. It works by calculating the probability of each class given a set of input features. The algorithm assumes that the input features are independent of each other, which is known as the "naive" assumption. Naïve Bayes is known for its simplicity and speed, making it a popular choice for text classification tasks.

The key concepts covered in this assignment would include:

* Understanding the different types of Naïve Bayes algorithms, such as Gaussian Naïve Bayes and Multinomial Naïve Bayes.
* Learning how to measure the performance of a Naïve Bayes model using metrics like accuracy and precision.
* Understanding how to handle missing values and skewed data when using Naïve Bayes.

Case study:- The dataset considered is named ‘data’. It has 569 rows and 33 columns. It contains data regarding the texture, area, radius and perimeter of cancer detected in a patient.

*#Importing the required libraries*

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

%matplotlib inline

data = pd.read\_csv('data.csv') data.shape

(569, 33)

data.head()

id diagnosis radius\_mean texture\_mean perimeter\_mean area\_mean \

0 842302 M 17.99 10.38 122.80

1001.0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 842517 | M | 20.57 | 17.77 | 132.90 |
| 1326.0 | |  |  |  |  |
| 2 84300903 | | M | 19.69 | 21.25 | 130.00 |
| 1203.0 | |  |  |  |  |
| 3 84348301 | | M | 11.42 | 20.38 | 77.58 |
| 386.1 | |  |  |  |  |
| 4 84358402 | | M | 20.29 | 14.34 | 135.10 |
| 1297.0 | |  |  |  |  |

smoothness\_mean compactness\_mean concavity\_mean concave points\_mean \

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0.11840 | 0.27760 | 0.3001 | |
| 0.14710 |  |  |  | |
| 1 | 0.08474 | 0.07864 | 0.0869 | |
| 0.07017 |  |  |  | |
| 2 | 0.10960 | 0.15990 | 0.1974 | |
| 0.12790 |  |  |  | |
| 3 | 0.14250 | 0.28390 | 0.2414 | |
| 0.10520 |  |  |  | |
| 4 | 0.10030 | 0.13280 | 0.1980 | |
| 0.10430 |  |  |  | |
| ... texture\_worst perimeter\_worst area\_worst smoothness\_worst \ | | | | |
| 0 ... | 17.33 | 184.60 | 2019.0 | 0.1622 |
| 1 ... | 23.41 | 158.80 | 1956.0 | 0.1238 |
| 2 ... | 25.53 | 152.50 | 1709.0 | 0.1444 |
| 3 ... | 26.50 | 98.87 | 567.7 | 0.2098 |

4 ... 16.67 152.20 1575.0 0.1374

compactness\_worst concavity\_worst concave points\_worst symmetry\_worst \

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0.6656 | 0.7119 | 0.2654 |
| 0.4601 |  |  |  |
| 1 | 0.1866 | 0.2416 | 0.1860 |
| 0.2750 |  |  |  |
| 2 | 0.4245 | 0.4504 | 0.2430 |
| 0.3613 |  |  |  |
| 3 | 0.8663 | 0.6869 | 0.2575 |
| 0.6638 |  |  |  |
| 4 | 0.2050 | 0.4000 | 0.1625 |
| 0.2364 |  |  |  |

fractal\_dimension\_worst Unnamed: 32

0 0.11890 NaN

1 0.08902 NaN

2 0.08758 NaN

3 0.17300 NaN

4 0.07678 NaN

[5 rows x 33 columns] data.describe()

id radius\_mean texture\_mean perimeter\_mean

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| area\_mean | \ | | | |
| count 5.690000e+02 | | 569.000000 | 569.000000 | 569.000000 |
| 569.000000 | |  |  |  |
| mean 3.037183e+07 | | 14.127292 | 19.289649 | 91.969033 |
| 654.889104 | |  |  |  |
| std 1.250206e+08 | | 3.524049 | 4.301036 | 24.298981 |

351.914129

min 8.670000e+03 6.981000 9.710000 43.790000

143.500000

25% 8.692180e+05 11.700000 16.170000 75.170000

420.300000

50% 9.060240e+05 13.370000 18.840000 86.240000

551.100000

75% 8.813129e+06 15.780000 21.800000 104.100000

782.700000

max 9.113205e+08 28.110000 39.280000 188.500000

2501.000000

smoothness\_mean compactness\_mean concavity\_mean concave points\_mean \

count 569.000000 569.000000 569.000000

569.000000

|  |  |  |
| --- | --- | --- |
| mean 0.096360 0.104341  0.048919  std 0.014064 0.052813 | 0.088799  0.079720 |  |
| 0.038803  min 0.052630 0.019380 | 0.000000 |
| 0.000000 |  |
| 25% 0.086370 0.064920 | 0.029560 |
| 0.020310 |  |
| 50% 0.095870 0.092630 | 0.061540 |
| 0.033500 |  |
| 75% 0.105300 0.130400 | 0.130700 |
| 0.074000  max 0.163400 0.345400 | 0.426800 |
| 0.201200 |  |
| symmetry\_mean ... texture\_worst | perimeter\_worst | area\_worst |
| \  count 569.000000 ... 569.000000 | 569.000000 | 569.000000 |
| mean 0.181162 ... 25.677223 | 107.261213 | 880.583128 |
| std 0.027414 ... 6.146258 | 33.602542 | 569.356993 |
| min 0.106000 ... 12.020000 | 50.410000 | 185.200000 |
| 25% 0.161900 ... 21.080000 | 84.110000 | 515.300000 |
| 50% 0.179200 ... 25.410000 | 97.660000 | 686.500000 |
| 75% 0.195700 ... 29.720000 | 125.400000 | 1084.000000 |
| max 0.304000 ... 49.540000 | 251.200000 | 4254.000000 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | smoothness\_worst | compactness\_worst | | concavity\_worst | \ |
| count | 569.000000 | 569.000000 | | 569.000000 |  |
| mean | 0.132369 | 0.254265 | | 0.272188 |  |
| std | 0.022832 | 0.157336 | | 0.208624 |  |
| min | 0.071170 | 0.027290 | | 0.000000 |  |
| 25% | 0.116600 | 0.147200 | | 0.114500 |  |
| 50% | 0.131300 | 0.211900 | | 0.226700 |  |
| 75% | 0.146000 | 0.339100 | | 0.382900 |  |
| max | 0.222600 | 1.058000 | | 1.252000 |  |
| concave points\_worst symmetry\_worst fractal\_dimension\_worst \ | | | | | |
| count | 569.000000 | | 569.000000 | 569.000000 | |
| mean | 0.114606 | | 0.290076 | 0.083946 | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| std |  | 0.065732 | 0.061867 | 0.018061 |
| min |  | 0.000000 | 0.156500 | 0.055040 |
| 25% |  | 0.064930 | 0.250400 | 0.071460 |
| 50% |  | 0.099930 | 0.282200 | 0.080040 |
| 75% |  | 0.161400 | 0.317900 | 0.092080 |
| max |  | 0.291000 | 0.663800 | 0.207500 |
| count | Unnamed: 32  0.0 |  | | |
| mean std min 25% | NaN NaN NaN NaN |
| 50% | NaN |
| 75%  max | NaN NaN |

[8 rows x 32 columns] data.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 569 entries, 0 to 568 Data columns (total 33 columns):

# Column Non-Null Count Dtype

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 id | 569 | non-null |  | int64 |
| 1 diagnosis | 569 | non-null |  | object |
| 2 radius\_mean | 569 | non-null |  | float64 |
| 3 texture\_mean | 569 | non-null |  | float64 |
| 4 perimeter\_mean | 569 | non-null |  | float64 |
| 5 area\_mean | 569 | non-null |  | float64 |
| 6 smoothness\_mean | 569 | non-null |  | float64 |
| 7 compactness\_mean | 569 | non-null |  | float64 |
| 8 concavity\_mean | 569 | non-null |  | float64 |
| 9 concave points\_mean | 569 | non-null |  | float64 |
| 10 symmetry\_mean | 569 | non-null |  | float64 |
| 11 fractal\_dimension\_mean | 569 | non-null |  | float64 |
| 12 radius\_se | 569 | non-null |  | float64 |
| 13 texture\_se | 569 | non-null |  | float64 |
| 14 perimeter\_se | 569 | non-null |  | float64 |
| 15 area\_se | 569 | non-null |  | float64 |
| 16 smoothness\_se | 569 | non-null |  | float64 |

|  |  |  |  |
| --- | --- | --- | --- |
| 17 compactness\_se | 569 | non-null | float64 |
| 18 concavity\_se | 569 | non-null | float64 |
| 19 concave points\_se | 569 | non-null | float64 |
| 20 symmetry\_se | 569 | non-null | float64 |
| 21 fractal\_dimension\_se | 569 | non-null | float64 |
| 22 radius\_worst | 569 | non-null | float64 |
| 23 texture\_worst | 569 | non-null | float64 |
| 24 perimeter\_worst | 569 | non-null | float64 |
| 25 area\_worst | 569 | non-null | float64 |
| 26 smoothness\_worst | 569 | non-null | float64 |
| 27 compactness\_worst | 569 | non-null | float64 |
| 28 concavity\_worst | 569 | non-null | float64 |
| 29 concave points\_worst | 569 | non-null | float64 |
| 30 symmetry\_worst | 569 | non-null | float64 |
| 31 fractal\_dimension\_worst | 569 | non-null | float64 |

32 Unnamed: 32 0 non-null float64 dtypes: float64(31), int64(1), object(1)

memory usage: 146.8+ KB

*#Dropping id column as it is not useful while diagnosing whether the patient has cancer or no*

data = data.drop(["id"], axis = 1)

*#Dropping Unnamed:32 column as it is not useful while diagnosing whether the patient has cancer or no*

data = data.drop(["Unnamed: 32"], axis = 1)

*#Dataset after dropping the columns*

data.head()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| diagnosis | radius\_mean | texture\_mean | perimeter\_mean | area\_mean | \ |
| 0 M | 17.99 | 10.38 | 122.80 | 1001.0 |  |
| 1 M | 20.57 | 17.77 | 132.90 | 1326.0 |  |
| 2 M | 19.69 | 21.25 | 130.00 | 1203.0 |  |
| 3 M | 11.42 | 20.38 | 77.58 | 386.1 |  |
| 4 M | 20.29 | 14.34 | 135.10 | 1297.0 |  |

smoothness\_mean compactness\_mean concavity\_mean concave points\_mean \

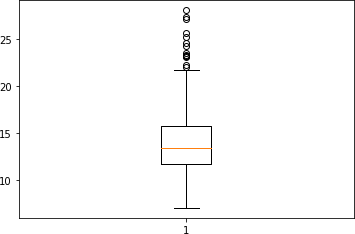
|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0.11840 | 0.27760 | 0.3001 |
| 0.14710 |  |  |  |
| 1 | 0.08474 | 0.07864 | 0.0869 |
| 0.07017 |  |  |  |
| 2 | 0.10960 | 0.15990 | 0.1974 |
| 0.12790 |  |  |  |
| 3 | 0.14250 | 0.28390 | 0.2414 |
| 0.10520 |  |  |  |
| 4 | 0.10030 | 0.13280 | 0.1980 |
| 0.10430 |  |  |  |

symmetry\_mean ... radius\_worst texture\_worst perimeter\_worst \

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0.2419 ... 25.38 | | | | 17.33 | 184.60 | |
| 1 | 0.1812 ... 24.99 | | | | 23.41 | 158.80 | |
| 2 | 0.2069 ... 23.57 | | | | 25.53 | 152.50 | |
| 3 | 0.2597 ... 14.91 | | | | 26.50 | 98.87 | |
| 4 | 0.1809 ... 22.54 | | | | 16.67 | 152.20 | |
|  | area\_worst | smoothness\_worst | | compactness\_worst | | concavity\_worst | \ |
| 0 | 2019.0 | 0.1622 | | 0.6656 | | 0.7119 |  |
| 1 | 1956.0 | 0.1238 | | 0.1866 | | 0.2416 |  |
| 2 | 1709.0 | 0.1444 | | 0.4245 | | 0.4504 |  |
| 3 | 567.7 | 0.2098 | | 0.8663 | | 0.6869 |  |
| 4 | 1575.0 | 0.1374 | | 0.2050 | | 0.4000 |  |
|  | concave points\_worst | | symmetry\_worst | | fractal\_dimension\_worst | | |
| 0 | 0.2654 | | 0.4601 | | 0.11890 | | |
| 1 | 0.1860 | | 0.2750 | | 0.08902 | | |
| 2 | 0.2430 | | 0.3613 | | 0.08758 | | |
| 3 | 0.2575 | | 0.6638 | | 0.17300 | | |
| 4 | 0.1625 | | 0.2364 | | 0.07678 | | |

[5 rows x 31 columns]

*#Checking outlires through boxplot* plt.boxplot(data['radius\_mean']) plt.show()



*#Checking Outlier by definition and treating outliers #getting median Age*

Age\_col\_df = pd.DataFrame(data['radius\_mean'])

Age\_median = Age\_col\_df.median()

*#getting IQR of Age column*

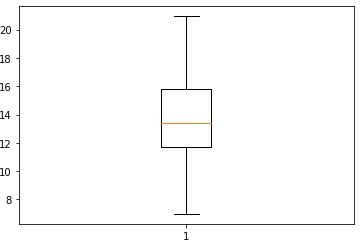
Q3 = Age\_col\_df.quantile(q=0.75) Q1 = Age\_col\_df.quantile(q=0.25)

IQR = Q3-Q1

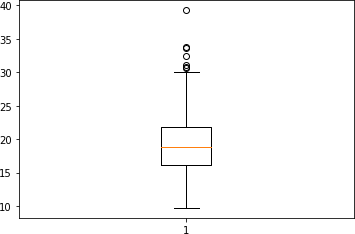
*#Deriving boundaries of Outliers* IQR\_LL = int(Q1 - 1.5\*IQR) IQR\_UL = int(Q3 + 1.5\*IQR)

*#Finding and treating outliers - both lower and upper end* data.loc[data['radius\_mean']>IQR\_UL , 'radius\_mean'] = int(Age\_col\_df.quantile(q=0.90)) data.loc[data['radius\_mean']<IQR\_LL , 'radius\_mean'] = int(Age\_col\_df.quantile(q=0.01))

*#Check max age value now* max(data['radius\_mean']) plt.boxplot(data['radius\_mean']) plt.show()



*#Checking outlires through boxplot* plt.boxplot(data['texture\_mean']) plt.show()



*#Checking Outlier by definition and treating outliers*

*#getting median Age*

Age\_col\_df = pd.DataFrame(data['texture\_mean']) Age\_median = Age\_col\_df.median()

*#getting IQR of Age column*

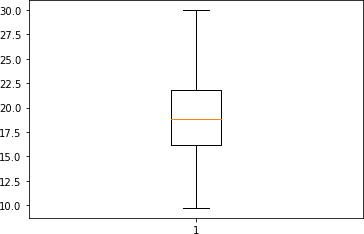
Q3 = Age\_col\_df.quantile(q=0.75) Q1 = Age\_col\_df.quantile(q=0.25)

IQR = Q3-Q1

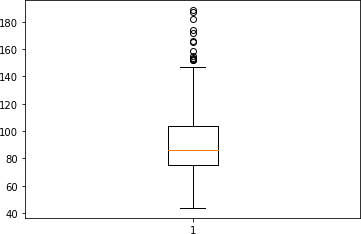
*#Deriving boundaries of Outliers* IQR\_LL = int(Q1 - 1.5\*IQR) IQR\_UL = int(Q3 + 1.5\*IQR)

*#Finding and treating outliers - both lower and upper end* data.loc[data['texture\_mean']>IQR\_UL , 'texture\_mean'] = int(Age\_col\_df.quantile(q=0.90)) data.loc[data['texture\_mean']<IQR\_LL , 'texture\_mean'] = int(Age\_col\_df.quantile(q=0.01))

*#Check max age value now* max(data['texture\_mean']) plt.boxplot(data['texture\_mean']) plt.show()



*#Checking outlires through boxplot* plt.boxplot(data['perimeter\_mean']) plt.show()



*#Checking Outlier by definition and treating outliers #getting median Age*

Age\_col\_df = pd.DataFrame(data['perimeter\_mean']) Age\_median = Age\_col\_df.median()

*#getting IQR of Age column*

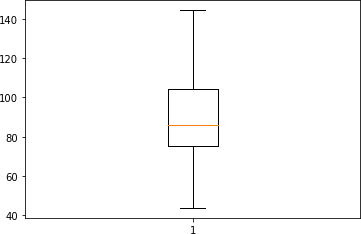
Q3 = Age\_col\_df.quantile(q=0.75) Q1 = Age\_col\_df.quantile(q=0.25)

IQR = Q3-Q1

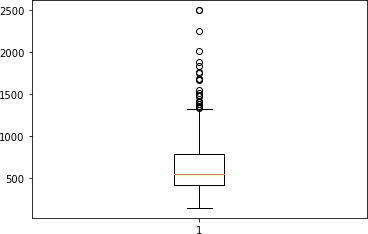
*#Deriving boundaries of Outliers* IQR\_LL = int(Q1 - 1.5\*IQR) IQR\_UL = int(Q3 + 1.5\*IQR)

*#Finding and treating outliers - both lower and upper end* data.loc[data['perimeter\_mean']>IQR\_UL , 'perimeter\_mean'] = int(Age\_col\_df.quantile(q=0.90)) data.loc[data['perimeter\_mean']<IQR\_LL , 'perimeter\_mean'] = int(Age\_col\_df.quantile(q=0.01))

*#Check max age value now* max(data['perimeter\_mean']) plt.boxplot(data['perimeter\_mean']) plt.show()



*#Checking outlires through boxplot* plt.boxplot(data['area\_mean']) plt.show()

*#Checking Outlier by definition and treating outliers*

*#getting median Age*

Age\_col\_df = pd.DataFrame(data['area\_mean']) Age\_median = Age\_col\_df.median()

*#getting IQR of Age column*

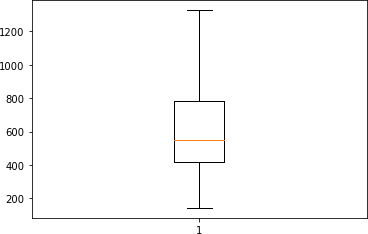
Q3 = Age\_col\_df.quantile(q=0.75) Q1 = Age\_col\_df.quantile(q=0.25)

IQR = Q3-Q1

*#Deriving boundaries of Outliers* IQR\_LL = int(Q1 - 1.5\*IQR) IQR\_UL = int(Q3 + 1.5\*IQR)

*#Finding and treating outliers - both lower and upper end* data.loc[data['area\_mean']>IQR\_UL , 'area\_mean'] = int(Age\_col\_df.quantile(q=0.90)) data.loc[data['area\_mean']<IQR\_LL , 'area\_mean'] = int(Age\_col\_df.quantile(q=0.01))

*#Check max age value now* max(data['area\_mean']) plt.boxplot(data['area\_mean']) plt.show()

feature\_cols = ['radius\_mean', 'perimeter\_mean', 'area\_mean'] X = data[feature\_cols]

Y = data['diagnosis']

**from** sklearn.naive\_bayes **import** GaussianNB

**from** sklearn.model\_selection **import** cross\_val\_score **from** sklearn.model\_selection **import** train\_test\_split **from** sklearn **import** metrics

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size = 0.3)

clf = GaussianNB()

clf = clf.fit(x\_train, y\_train) y\_pred = clf.predict(x\_test)

print("Accuracy:", metrics.accuracy\_score(y\_test, y\_pred))*#Testing accuracy*

Accuracy: 0.8771929824561403

Conclusion:- In conclusion, this assignment demonstrates the importance of Naïve Bayes in classification. By understanding the theory behind Naïve Bayes and how to apply it to a dataset, we can make accurate predictions and gain insights from the data.